7. Standard extensions

7.1. Image extension

7.2. ASCII table extension

7.3. Binary table extension

7.4. Foreign extension

The foreign extension provides a mechanism for encapsulating an arbitrary file or direcotory tree of files in a *FITS* file, where each foreign extension contains a single file. A set of "file group" keywords are also defined which store various computer system attributes of the original file and provide a means for associating a group of related *FITS* extensions of any type. This enables an entire directory tree of files (containing both *FITS* and non-*FITS* files) to be archived in a single multi-extension *FITS* file. That *FITS* file can then be used to restore an exact copy of the original directory tree of files.

The foreign extension was initially developed by the IRAF computer programming group primarily for use within the National Optical Astronomical Observatory (NOAO) High Performance Pipeline System. In particular, support for reading and writing foreign extensions has been implemented in the *fgread* and *fgwrite* utility programs in the *fitsutil* external IRAF software package. The user documentation provided with those utility programs contain useful examples and practical guidelines related to the use of foreign extensions in *FITS* files.

7.4.1. Mandatory keywords

The `XTENSION` keyword is required to be the first keyword of all foreign extensions. The keyword records in the header of a foreign extension *must* use the keywords defined in Table 1 in the order specified. No other keywords *may* intervene between the `XTENSION` and `GCOUNT` keywords.

The optional `EXTNAME` keyword is used only to identify the extension within the *FITS* file.

7.4.2. File Group (FG) Keywords

In order to be able to unpack foreign extensions and restore the original files, a set of "FG" keywords are defined here to store various system attributes about each file. Note that some of the currently defined keywords only apply to files on UNIX computer systems and may not have an exact counterpart under other operating systems such as Windows.

`FG_GROUP` keyword. The value field *shall* contain a character string containing the name of the group to which the current file belongs. The group name is arbitrary (like a filename) and is assigned by the user when the group of files is written. For example, a group name for a directory tree might be the name of the root directory. It is up to the writer program to assign a unique group name if the user does not predefine one.

`FG_FNAME` keyword. The value field *shall* contain a character string containing the filename of the file associated with the current extension. The maximum filename length is 67 characters. Any printable character except apostrophe is permitted. For an extension of type foreign where the file type is directory, `FG_FNAME` is the name of the directory.

`FG_FTYPE` keyword. The value field *shall* contain a character string that describes the physical file type. The following types are recognized:

- `'text'` - A file containing only text. Stored 8 bits per character using newline to delimit lines of text (like Unix).
- `'binary'` - Any file which is not a text file or one of the known file types. Stored as a byte stream without any conversion.
- `'directory'` - implementation dependent
- `'symlink'` - implementation dependent
- `'FITS'` - a native FITS extension
- `'FITS-MEF'` - a native multi-extension FITS (MEF) file. No count of the number of extensions in the MEF file is given, rather the MEF group consists of all subsequent extensions until a *FITS* extension is encountered which starts a new file.

`FG_LEVEL` keyword. The value field *shall* contain an integer that specifies the directory nesting level. All of the files in a directory are at the same level. Foreign extensions of type directory are used to name the directories at each level so that path names can be reconstructed (this scheme assumes that the extensions in a file group are ordered). Level 0 (zero) is the root directory of the file group. The root directory is unnamed and is implicitly the user's current working directory into which the file in the foreign extension would be unpacked. When packing files into foreign *FITS* extensions, the current working directory could be a logical choice for the `FG_GROUP` file group name.

`FG_FSIZE` keyword. The value field *shall* contain an integer that specifies the size in bytes of the data portion of the file. This value is always identical to the value of the `PCOUNT` keyword. In the case of a file with a `FG_TYPE` value equal to `'directory'`, the `FG_FSIZE` value is zero.

`FG_MTYPE` optional keyword. The value field *shall* contain a character string that contains the logical or "mime" type of the file.

`FG_FMODE` keyword. The value field *shall* contain a character string that specifies the file 'mode' on Unix operating systems (e.g., `'rwx-rwx-rwx'`, where bits not set are given as `'-'`).

`FG_FUOWN` keyword. The value field *shall* contain a character string that specifies the Unix user ID (UID) of the file.

`FG_FUGRP` keyword. The value field *shall* contain a character string that specifies the Unix group ID (GID) of the file.

`FG_CTIME` keyword. The value field *shall* contain a character string that specifies the file creation time as a UTC value expressed as an ISO 8601 string.

Table 1. Mandatory keywords in foreign extensions.

#	Keyword
1	XTENSION=␣'FOREIGN␣'
2	BITPIX = 8
3	NAXIS = 0
4	PCOUNT = file size in bytes
5	GCOUNT = 1
	⋮
	(other keywords, including . . .)
	EXTNAME = 'filename' (*optional*)
	⋮
last	END

FG_MTIME keyword. The value field *shall* contain a character string that specifies the file modification time as a UTC value expressed as an ISO 8601 string.

FG_COMP keyword. This keyword is reserved for possible future use in cases where the file is also compressed as it is written to the foreign extension..